Software Managed Networks
Portfolio Task – C-Lab-Report - 2

Leo Holden
Dept. Of Computing Technologies
Swinburne University of Technology
Hawthorn, Australia
103996982@student.swin.edu.au

I. Aim

The aim of this lab report is to consolidate the information and skills learnt regarding Mininet, Open Virtual Switch (OVS) and Ryu controller setup. A virtual network will be established and various commands will be used to configure the network routing capabilities. Further analysis is conducted to understand the results using OVS and Mininet Command Line Interface (CLI) commands.

II. Equipment

The virtualized environment is produced through VMware creating a Linux virtual machine (VM) preinstalled with relevant tools. Running a 64-bit Ubuntu version 18.04.5 LTS, the VM has been allocated the following hardware resources: 8 GB RAM, 6 cores and a 40 GB disk partition. VM has Mininet 2.3.0 and Ryu 4.1.5 installed. *Table 1* shows the specification of the local machine and VM. The VM should be capable of operating at minimal hardware allocations e.g. 2 GB of RAM and 2 cores as Mininet is lightweight. (Note: Mininet requires a Linux operating system to emulate networking devices)

Specification	**Local Machine**	**Virtual Machine**
CPU	12th Gen Intel i5-12500H (12 cores / 16 threads)	6 virtual CPUs allocated
RAM	16 GB	8 GB
Storage	512 GB	40 GB
Operating System	Windows 11 64-bit	Ubuntu 18.04.5 LTS 64-bit
Virtualisation Tool	VMware Workstation 17	
Software Versions		Mininet 2.3.0, Ryu 4.1.5

Table 1 Local hardware and virtual machine's specifications

III. Method

As per lab handout: P-Lab-05-SDN-Controller-Advanced

IV. Results

This lab is broken into three sections that outlines what is performed which are sections: A. Setup the network, B. L3 Routing with Ryu, C. L3 Routing with Ryu Exercise.

A. Setup the Network

Mininet is used to construct a linear virtual network consisting of a controller (c0), two switches (s1, s2) and two hosts (h1, h2). The command: "`sudo mn --controller remote,ip=127.0.0.1 --topo linear,2`" is used to create the network, *Figure 1* shows the addressing scheme.

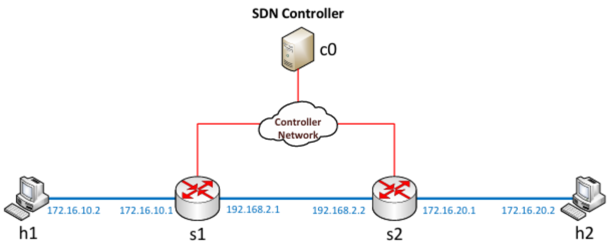
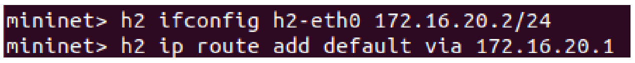
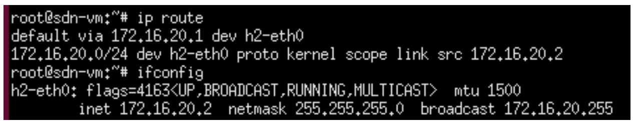


Figure 1 Network topology [1]



```
mininet> h2 ifconfig h2-eth0 172.16.20.2/24
mininet> h2 ip route add default via 172.16.20.1
```

Figure 2 Mininet commands to configure IP addressing



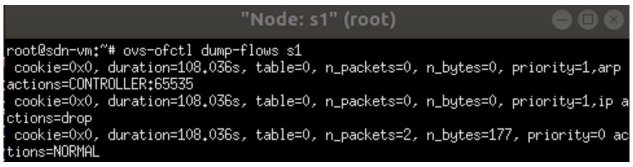
```
root@sdn-vm:~# ip route
default via 172.16.20.1 dev h2-eth0
172.16.20.0/24 dev h2-eth0 proto kernel scope link src 172.16.20.2
root@sdn-vm:~# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.20.2  netmask 255.255.255.0  broadcast 172.16.20.255
```

Figure 3 xterm h2 IP addressing verification

Figure 2 shows the example for h2 Mininet commands that configure the hosts IP and default gateway. The commands should be issued for each host, *Figure 3* verifies the configurations

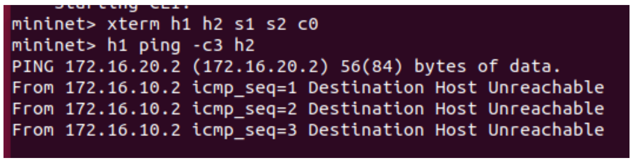
B. L3 Routing with Ryu

Redirecting too the appropriate directory containing the Ryu controller application and executing the following command: "`ryu-manager--verbose rest_router.py gui_topology/gui_topology.py`". Viewing the flow dump as seen in *Figure 4* verifies OVS functionality and that the Ryu controller is in operation.



```
root@sdn-vm:~# ovs-ofctl dump-flows s1
 cookie=0x0, duration=108.036s, table=0, n_packets=0, n_bytes=0, priority=1,arp
actions=CONTROLLER:65535
 cookie=0x0, duration=108.036s, table=0, n_packets=0, n_bytes=0, priority=1,ip a
ctions=drop
 cookie=0x0, duration=108.036s, table=0, n_packets=2, n_bytes=177, priority=0 ac
tions=NORMAL
```

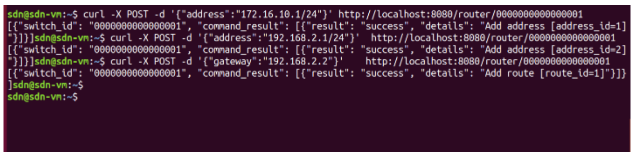
Figure 4 s1 flow dump



```
mininet> xterm h1 h2 s1 s2 c0
mininet> h1 ping -c3 h2
PING 172.16.20.2 (172.16.20.2) 56(84) bytes of data.
From 172.16.10.2 icmp_seq=1 Destination Host Unreachable
From 172.16.10.2 icmp_seq=2 Destination Host Unreachable
From 172.16.10.2 icmp_seq=3 Destination Host Unreachable
```

Figure 5 Ping attempt after Ryu controller setup

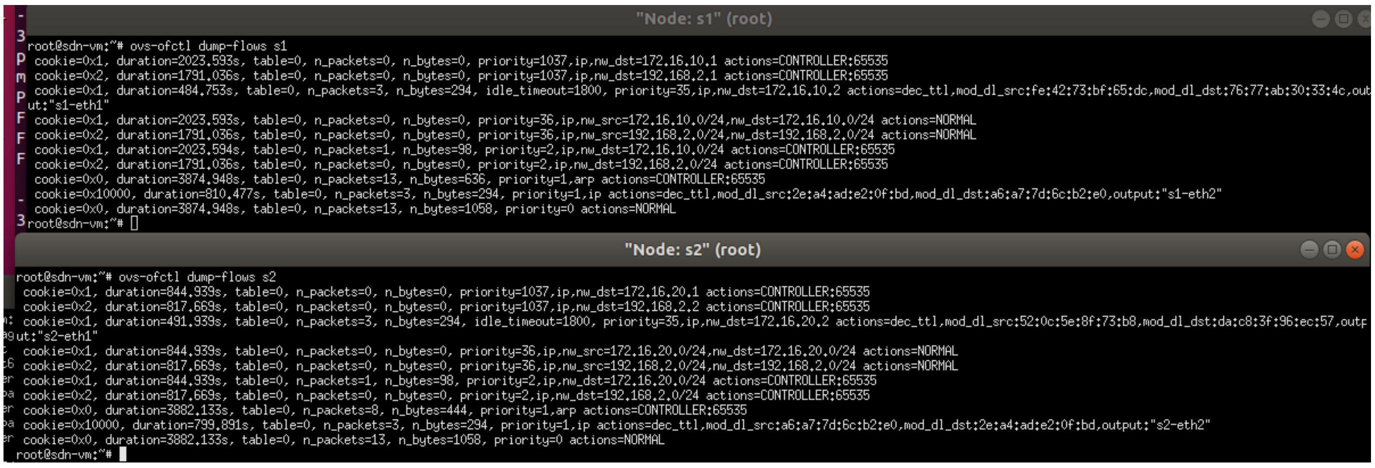
The following flows do not permit layer 3 connectivity and can be demonstrated by issuing a ping between hosts. To establish layer 3 connectivity, the necessary interface configurations must be made for each switch.



```
sdn@sdn-vm:~$ curl -X POST -d '{"address":"172.16.10.1/24"}' http://localhost:8080/router/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Add address [address_id=1]
"}]}]sdn@sdn-vm:~$ curl -X POST -d '{"address":"192.168.2.1/24"}'  http://localhost:8080/router/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Add address [address_id=2]
"}]}]sdn@sdn-vm:~$ curl -X POST -d '{"gateway":"192.168.2.2"}'    http://localhost:8080/router/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Add route [route_id=1]"}]}
]sdn@sdn-vm:~$
sdn@sdn-vm:~$
```

Figure 6 Configure s1 virtual interfaces and gateway for L3 routing

Once these flows are verified to be installed on each switch, a ping from h1 and h2 will successfully complete. To verify this, the command as seen in *Figure 5* can be executed to verify layer 3 connectivity. (Command in Mininet CLI: "`h1 ping -c3 h2`"). Verification of the virtual interfaces for each switch is viewed by the command: "`curl http://localhost:8080/router/<switch_id>`". The expected result is a 200 OK return value, stating the configured virtual interfaces and default route to the gateway.

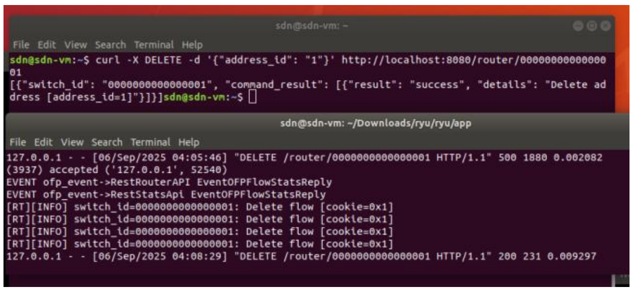


```
root@sdn-vm:~# ovs-ofctl dump-flows s1
 cookie=0x1, duration=2023.593s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_dst=172.16.10.1 actions=CONTROLLER:65535
 cookie=0x2, duration=1791.036s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_dst=192.168.2.1 actions=CONTROLLER:65535
 cookie=0x1, duration=484.753s, table=0, n_packets=3, n_bytes=294, idle_timeout=1800, priority=35,ip,nw_dst=172.16.10.2 actions=dec_ttl,mod_dl_src:fe:42:73:bf:65:dc,mod_dl_dst:76:77:ab:30:33:4c,out
ut:"s1-eth1"
 cookie=0x1, duration=2023.593s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=172.16.10.0/24,nw_dst=172.16.10.0/24 actions=NORMAL
 cookie=0x2, duration=1791.036s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=192.168.2.0/24,nw_dst=192.168.2.0/24 actions=NORMAL
 cookie=0x1, duration=2023.594s, table=0, n_packets=1, n_bytes=98, priority=2,ip,nw_dst=172.16.10.0/24 actions=CONTROLLER:65535
 cookie=0x2, duration=1791.036s, table=0, n_packets=0, n_bytes=0, priority=2,ip,nw_dst=192.168.2.0/24 actions=CONTROLLER:65535
 cookie=0x0, duration=3874.948s, table=0, n_packets=13, n_bytes=636, priority=1,arp actions=CONTROLLER:65535
 cookie=0x10000, duration=810.477s, table=0, n_packets=3, n_bytes=294, priority=1,ip actions=dec_ttl,mod_dl_src:2e:a4:ad:e2:0f:bd,mod_dl_dst:a6:a7:7d:6c:b2:e0,output:"s1-eth2"
 cookie=0x0, duration=3874.948s, table=0, n_packets=13, n_bytes=1058, priority=0 actions=NORMAL
root@sdn-vm:~# 
```

```
root@sdn-vm:~# ovs-ofctl dump-flows s2
 cookie=0x1, duration=844.939s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_dst=172.16.20.1 actions=CONTROLLER:65535
 cookie=0x2, duration=817.669s, table=0, n_packets=0, n_bytes=0, priority=1037,ip,nw_dst=192.168.2.2 actions=CONTROLLER:65535
 cookie=0x1, duration=491.939s, table=0, n_packets=3, n_bytes=294, idle_timeout=1800, priority=35,ip,nw_dst=172.16.20.2 actions=dec_ttl,mod_dl_src:52:0c:5e:8f:73:b8,mod_dl_dst:da:c8:3f:96:ec:57,outp
ut:"s2-eth1"
 cookie=0x1, duration=844.939s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=172.16.20.0/24,nw_dst=172.16.20.0/24 actions=NORMAL
 cookie=0x2, duration=817.669s, table=0, n_packets=0, n_bytes=0, priority=36,ip,nw_src=192.168.2.0/24,nw_dst=192.168.2.0/24 actions=NORMAL
 cookie=0x1, duration=844.939s, table=0, n_packets=1, n_bytes=98, priority=2,ip,nw_dst=172.16.20.0/24 actions=CONTROLLER:65535
 cookie=0x2, duration=817.669s, table=0, n_packets=0, n_bytes=0, priority=2,ip,nw_dst=192.168.2.0/24 actions=CONTROLLER:65535
 cookie=0x0, duration=3882.133s, table=0, n_packets=8, n_bytes=444, priority=1,arp actions=CONTROLLER:65535
 cookie=0x10000, duration=799.891s, table=0, n_packets=3, n_bytes=294, priority=1,ip actions=dec_ttl,mod_dl_src:a6:a7:7d:6c:b2:e0,mod_dl_dst:2e:a4:ad:e2:0f:bd,output:"s2-eth2"
 cookie=0x0, duration=3882.133s, table=0, n_packets=13, n_bytes=1058, priority=0 actions=NORMAL
root@sdn-vm:~# 
```

Figure 7 s1 & s2 flow dump verifying new flows

To remove a virtual interface another, curl command can be executed as seen in *Figure 8,* the respective verification of viewing the controller's output terminal can be seen too. The address_id value can be determined through the aforementioned curl command for verification of virtual interfaces for switches. Further information about Ryu rest router can be found in [2].



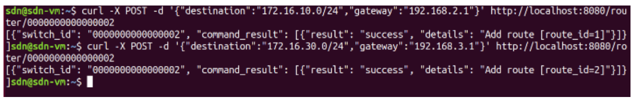
```
sdn@sdn-vm:~$ curl -X DELETE -d '{"address_id": "1"}' http://localhost:8080/router/00000000000000
01
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Delete ad
dress [address_id=1]"}]}]sdn@sdn-vm:~$ 
```

```
127.0.0.1 - - [06/Sep/2025 04:05:46] "DELETE /router/0000000000000001 HTTP/1.1" 500 1880 0.002082
(3937) accepted ('127.0.0.1', 52540)
EVENT ofp_event->RestRouterAPI EventOFPFlowStatsReply
EVENT ofp_event->RestStatsApi EventOFPFlowStatsReply
[RT][INFO] switch_id=0000000000000001: Delete flow [cookie=0x1]
[RT][INFO] switch_id=0000000000000001: Delete flow [cookie=0x1]
[RT][INFO] switch_id=0000000000000001: Delete flow [cookie=0x1]
[RT][INFO] switch_id=0000000000000001: Delete flow [cookie=0x1]
127.0.0.1 - - [06/Sep/2025 04:08:29] "DELETE /router/0000000000000001 HTTP/1.1" 200 231 0.009297
```

Figure 8 Deletion of virtual interface of address_id 1

C. L3 Routing with Ryu Exercise

This section proceeds with using the same linear network topology with an additional switch and host. However, to configure the default routes on s2 to s1 and s3 (Note: s1-s2-s3 is how the switches are connected), a different command is used as seen in *Figure 9.* This command maps IPs in specific subnet to specific IP address (static route). The remaining configurations are the same as previous with the last topology but with an additional switch and host.



```
sdn@sdn-vm:~$ curl -X POST -d '{"destination":"172.16.10.0/24","gateway":"192.168.2.1"}' http://localhost:8080/rou
ter/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "Add route [route_id=1]"}]}
]sdn@sdn-vm:~$ curl -X POST -d '{"destination":"172.16.30.0/24","gateway":"192.168.3.1"}' http://localhost:8080/ro
ter/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "Add route [route_id=2]"}]}
]sdn@sdn-vm:~$ 
```

Figure 9 Configuration for s2 default routes

V. Discussion

This section provides an informative analysis of major steps in each section of the lab. It explains how the lab questions were answered.

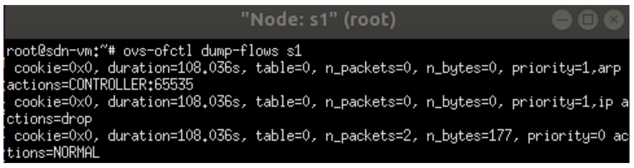
A. Setup the Network

This section involved establishing the Mininet network and configuring the IP addressing according to the network topology outlined in *Figure 1.* The verification commands can be seen in *Figure 3* by creating a virtual terminal (xterm) for each host and executing the `ifconfig` and `ip route`

CLI command. (Note: when instantiating the Mininet network, the parameter “2” controls the number of nodes and therefore in subsection C, 3 is used)

B. L3 Routing with Ryu

This section configures virtual interfaces and default gateways on switches through the Ryu controller by utilizing the Rest router API. This in turn configures layer 3 routing capabilities and therefore h1 and h2 connectivity.



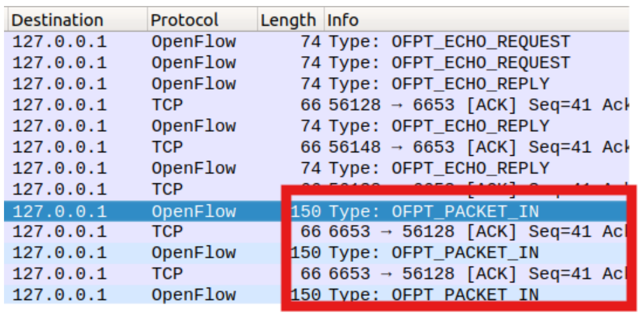
```
root@sdn-vm:~# ovs-ofctl dump-flows s1
 cookie=0x0, duration=108.036s, table=0, n_packets=0, n_bytes=0, priority=1,arp
actions=CONTROLLER:65535
 cookie=0x0, duration=108.036s, table=0, n_packets=0, n_bytes=0, priority=1,ip a
ctions=drop
 cookie=0x0, duration=108.036s, table=0, n_packets=2, n_bytes=177, priority=0 ac
tions=NORMAL
```

Figure 10 Initial flows install upon Ryu controller setup

The flows created allow for ARP requests to be forwarded to the controller for routing decisions and create flows, all other IP type packets will be dropped until a higher priority flow is installed. The fall-back flow will resort to layer 2 forwarding. A higher priority number indicates priority over other flows and action NORMAL indicates layer 2 forwarding. In the case of this flow being used, controller interaction is not necessary whereas the action in the first flow for ARP packets is sent to the controller indicated by action=CONTROLLER. Therefore, when attempting to ping from each host located in different subnets, it will not succeed. As before the ICMP packets are sent, an ARP request is required to map the MAC and IP address for the packet header. However, because there are no interfaces on the switches designated as the default gateways for each host, ARP requests will have no return values. Furthermore, because the hosts are in different subnets, the fall back to layer 2 forwarding will also fail. To enable layer 3 connectivity the Ryu REST router API can be interacted with the curl commands as seen in *Figure 6*. The flow dump can be analysed in *Figure 7* which shows newly installed flows for s1 and s2.

C. L3 Routing with Ryu Exercise

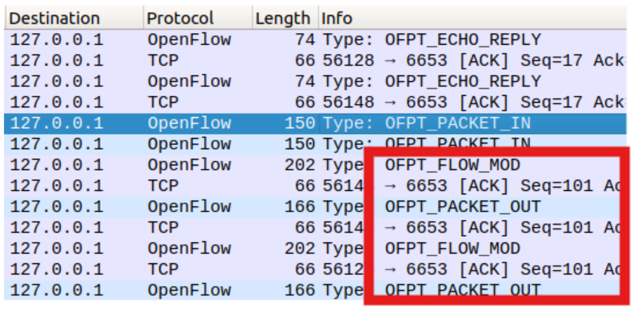
The result of interacting with the Ryu controller API is creating virtual interfaces and route table building for the switches. This allows for the ARP requests to complete for when h1 attempts an ARP request for its default gateway. Wireshark captures of before and after can demonstrate the OpenFlow protocol attempt to resolve ARP requests.



```
Destination    Protocol    Length Info
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REQUEST
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REQUEST
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REPLY
127.0.0.1      TCP             66 56128 → 6653 [ACK] Seq=41 Ack
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REPLY
127.0.0.1      TCP             66 56148 → 6653 [ACK] Seq=41 Ack
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REPLY
127.0.0.1      TCP
127.0.0.1      OpenFlow       150 Type: OFPT_PACKET_IN
127.0.0.1      TCP             66 6653 → 56128 [ACK] Seq=41 Ac
127.0.0.1      OpenFlow       150 Type: OFPT_PACKET_IN
127.0.0.1      TCP             66 6653 → 56128 [ACK] Seq=41 Ac
127.0.0.1      OpenFlow       150 Type: OFPT_PACKET_IN
```

Figure 11 Ryu controller Wireshark capture

Three attempts to ICMP ping from h1 to h2 was executed, *Figure 11* shows three OFPT_PACKET_IN OpenFlow protocol packets. These packets are contacting the Ryu controller on how to handle these packets, with proper configuration OFPT_PACKET_OUT packets are transmitted in response which is the immediate response to how to handle the packet. Following this an OFPT_FLOW_MOD is sent to insert a flow that will handle future similar packets.



```
Destination    Protocol    Length Info
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REPLY
127.0.0.1      TCP             66 56128 → 6653 [ACK] Seq=17 Ack
127.0.0.1      OpenFlow        74 Type: OFPT_ECHO_REPLY
127.0.0.1      TCP             66 56148 → 6653 [ACK] Seq=17 Ack
127.0.0.1      OpenFlow       150 Type: OFPT_PACKET_IN
127.0.0.1      OpenFlow       150 Type: OFPT_PACKET_IN
127.0.0.1      OpenFlow       202 Type  OFPT_FLOW_MOD
127.0.0.1      TCP             66 5614  → 6653 [ACK] Seq=101 Ac
127.0.0.1      OpenFlow       166 Type  OFPT_PACKET_OUT
127.0.0.1      TCP             66 5614  → 6653 [ACK] Seq=101 Ac
127.0.0.1      OpenFlow       202 Type  OFPT_FLOW_MOD
127.0.0.1      TCP             66 5612  → 6653 [ACK] Seq=101 Ac
127.0.0.1      OpenFlow       166 Type  OFPT_PACKET_OUT
```

Figure 12 Ryu controller Wireshark capture after API configuration

Once the appropriate configurations are made by interacting with the Ryu controller API, as seen in *Figure 6*. As seen in *Figure 12,* the described packets appear, and as a result performing a flow dump on s1 and s2 will result in a similar output to what is seen in *Figure 7.* These new flows therefore allow layer 3 connectivity and successful ICMP pings.

VI. Conclusion

In conclusion to this lab report SDN is demonstrated using Ryu controller which handles the control plane logic which orchestrates routing policies for layer 3 communication. A virtual network is deployed quickly to test the Ryu controller through Mininet and procedure of establishing layer 3 connectivity has been complete. In summary, when using the Ryu controller, the Ryu API allows for configuration through curl commands which establishes virtual interfaces and configures routing information to which OpenFlow protocol packets and be exchanged to install appropriate flows. Resulting in the switches being able to handle packets destined outside their subnet, along with flows which will handle other packets such as ARP or if packet destination is within the subnet, layer 2 forwarding will take effect without Ryu controller intervention.

VII. References

[1] Swinburne University of Technology, *P-Lab-05: SDN Controller Advanced Lab*, unpublished teaching material, 2025 [Accessed: Sept. 19, 2025]

[2] RYU Project Team, “Router — rest_router,” *Ryubook 1.0 Documentation*, 2014. [Online]. Available: https://osrg.github.io/ryu-book/en/html/rest_router.html [Accessed: Sept. 19, 2025]

[3] Open vSwitch, “ovs-ofctl(8) — administer OpenFlow switches,” *Linux manual pages*, man7.org. [Online]. Available: https://www.man7.org/linux/man-pages/man8/ovs-ofctl.8.html [Accessed: Sept. 22, 2025]

[4] Open Networking Foundation, *“OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04),”* ONF, ONF TS-006, June 25, 2012. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf [Accessed: Sept. 22, 2025]